ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

Тема: Комплексна система управління сервісами з використанням засобів контейнеризації

Підготував:
Студент 5 курсу групи КСЗ-51
Медведенко Олександр
Науковий керівник:
Світлана Куфтеріна

Київ 2024

Актуальність

Сучасний бізнес у цифрову епоху потребує ефективного управління IT-сервісами для підвищення продуктивності команди та якості продуктів. Технології контейнеризації, як-то Docker, забезпечують ізоляцію сервісів та їх гнучкість. Розробка системи управління IT-сервісами на основі контейнеризації підвищує ефективність та конкурентоспроможність бізнесу.

Мета

Мета: Розробка та впровадження комплексної системи управління IT-сервісами з використанням засобів контейнеризації для підвищення ефективності, надійності та масштабованості розробки та розгортання програмного забезпечення.

Об'єкт

Об'єкт роботи: Процеси управління IT-сервісами в контексті сучасних розподілених систем та мікросервісної архітектури.

Предмет

Предмет роботи: Методи та інструменти контейнеризації, зокрема Docker, для побудови та управління комплексною системою IT-сервісів.

Завдання дослідження

1.Аналіз існуючих систем управління IT-сервісами та засобів контейнеризації.
2.Розробка моделі управління процесами для дистанційної команди.
3.Проектування та реалізація системи управління IT-сервісами з використанням Docker.
4.Апробація та оцінка ефективності запропонованої системи.

Для вирішення поставлених завдань в теоретичній частині роботи було:

1.Визначено поняття "сервіс" та його роль в сучасному IT-ландшафті.
2.Досліджено історичний контекст та еволюцію управління IT-сервісами.
3.Проаналізовано актуальні патенти та наукові публікації з теми дослідження.
4.Розглянуто ключові архітектурні підходи до побудови веб-додатків: монолітна, SOA, подійно-орієнтована та мікросервісна.

Мікросервісна архітектура та інструменти

Використано шаблони проектування Strangler, Bulkhead та Sidecar для забезпечення поступової міграції, стійкості до відмов та розширення функціональності.

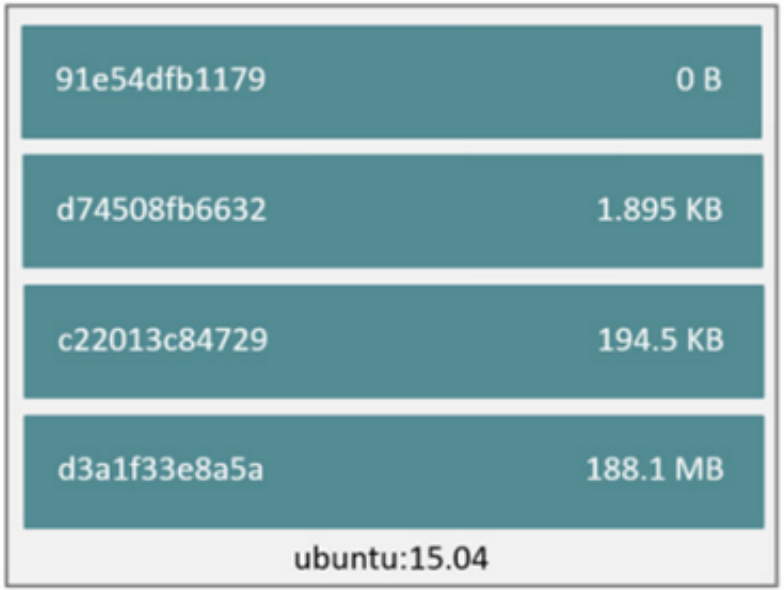
Обрано Docker як платформу контейнеризації для ізоляції та ефективного управління мікросервісами.

Розгортання та забезпечення надійності

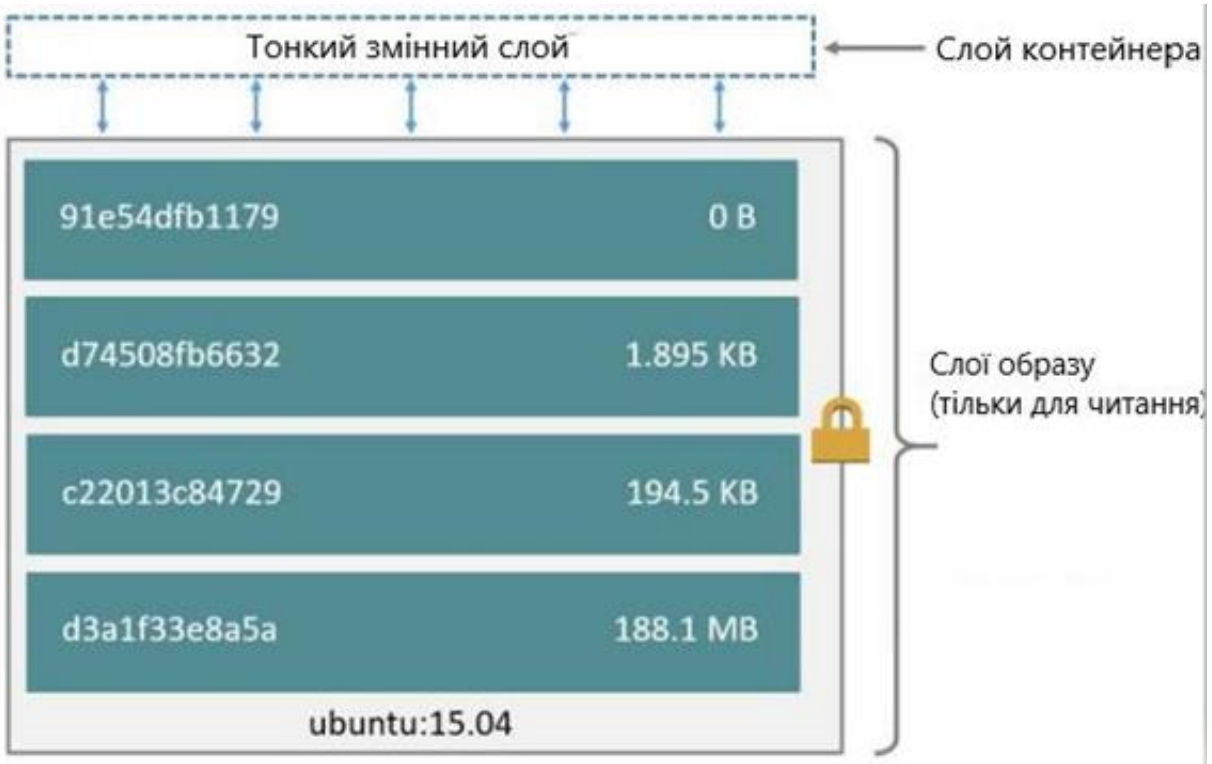
Проаналізовано різні стратегії розгортання мікросервісів: на одному сервері, віртуалізація та контейнеризація.

Обрано контейнеризацію як оптимальний підхід, враховуючи баланс між ізоляцією, ефективністю використання ресурсів та гнучкістю.

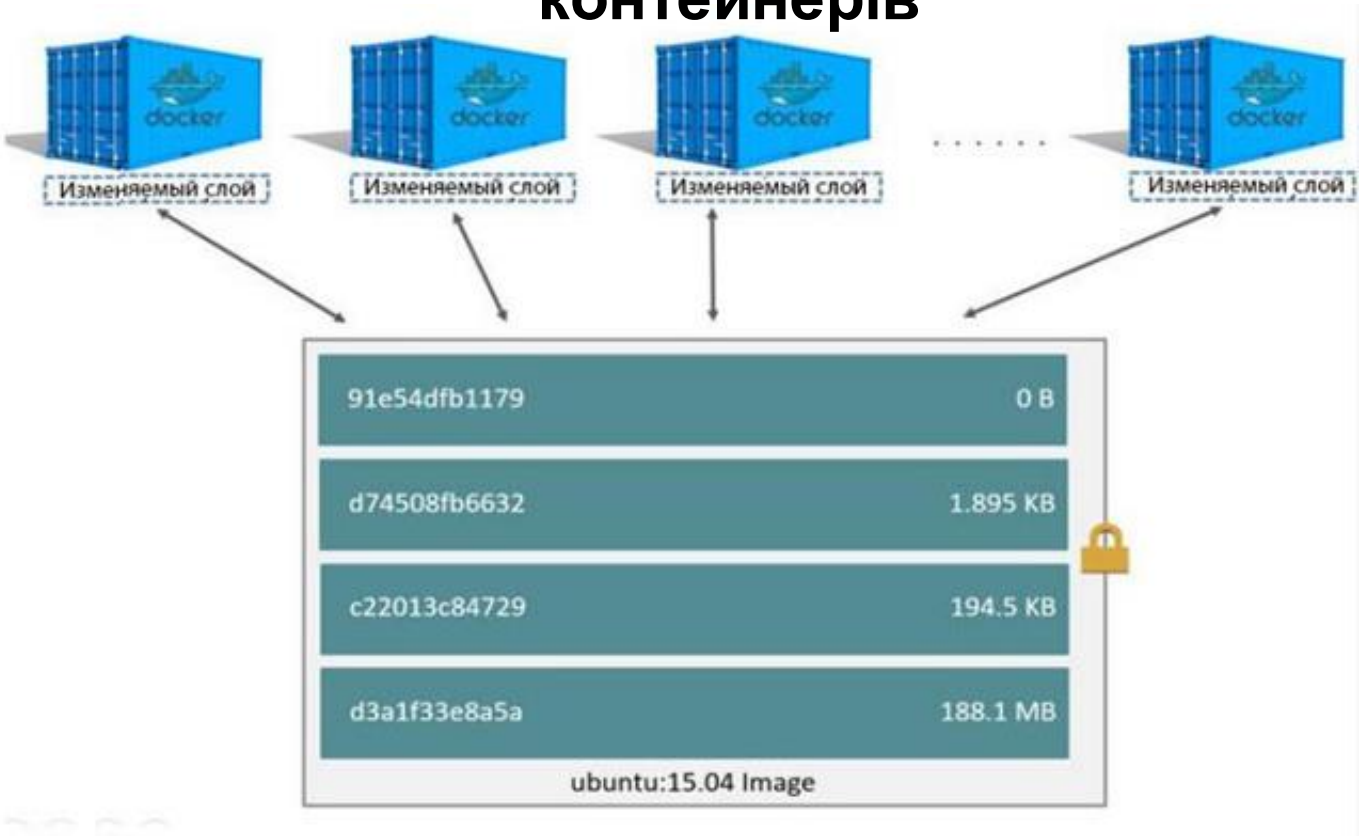
Рівні образу «Docker»



Шари запущеного контейнера



Використання одного образу безліччю контейнерів



Інтерфейс взаємодії із сервісом users-api

```
+--------+----------+------------------+------+-----------------------------------------------+------------+
| Domain | Method   | URI              | Name | Action                                        | Middleware |
+--------+----------+------------------+------+-----------------------------------------------+------------+
|        | GET|HEAD | api/v1/users     |      | App\Http\Controllers\UsersController@index    |            |
|        | GET|HEAD | api/v1/users/{id} |      | App\Http\Controllers\UsersController@getById |            |
+--------+----------+------------------+------+-----------------------------------------------+------------+
```

Приклад відповіді на API запит


```
{
    "data": [
        {
            "id": 1,
            "name": "magnus.bashirian",
            "email": "gislason.loy@gmail.com",
            "created_at": "2020-01-19 13:22:17",
            "updated_at": null
        },
    ],
    "meta": {
        "container_id": "6ae6484f505a",
        "service_name": "users-api"
    }
}
```

Інтерфейс взаємодії із сервісом documents-api

Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	api/v1/documents		App\Http\Controllers\DocumentsController@index	
	GET|HEAD	api/v1/documents/user/{id}		App\Http\Controllers\DocumentsController@getByUserId	
	GET|HEAD	api/v1/documents/{id}		App\Http\Controllers\DocumentsController@getById	

Інтерфейс взаємодії із сервісом statistics-api

Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	api/v1/statistics		App\Http\Controllers\StatisticsController@index	
	GET|HEAD	api/v1/statistics/user/{id}		App\Http\Controllers\StatisticsController@getByUserId	

Конфігурація автоматизованих збірок

Розгортання та забезпечення надійності

Застосовано Docker Swarm для кластеризації та забезпечення високої доступності системи.

Впроваджено моніторинг системи за допомогою Prometheus та Grafana для збору та візуалізації метрик продуктивності.

Аналіз тривалості збірки образів

Сервіс	Кількість вимірів	Розмір образу (мегабайт)	Середня тривалість збірки (секунд)
users-api	20	204,7	195,4
documents-api	20	201,4	191,8
statistics-api	20	203,1	194,3
api-gateway	20	48,5	54,2

Аналіз часу очікування відповіді.

Сервіс	Тривалість, секунди
Api Gateway	0,19
Users-api	0,38
Documents-api	0,22
Statistics-api	0,3
Усього	1,09

Список віртуальних машин

```
NAME            ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER      ERRORS
default         *        virtualbox   Running   tcp://192.168.99.100:2376           v19.03.5
swarm-worker    -        virtualbox   Stopped                                       Unknown
swarm-worker-2  -        virtualbox   Stopped                                       Unknown
swarm-worker-3  -        virtualbox   Stopped                                       Unknown
```

Список сервісів кластеру

```
docker@default: $ docker service ls
ID                  NAME                MODE                REPLICAS
r8mazuiasx87        app_api-gateway     replicated          1/1
qk93dfs6qpps        app_documents-api   replicated          3/3
ooapjb9mdzbj        app_documents-db    replicated          1/1
cidkfcovzlnp        app_statistics-api  replicated          3/3
y65uludclat8        app_statistics-db   replicated          1/1
nxrpk81hhzcl        app_users-api       replicated          3/3
biiqm82dej2h        app_users-db        replicated          1/1
```

Забезпечення відмово-стійкості сервісів

ID	NAME	NODE	DESIRED STATE	CURRENT STATE
intzhwvqw282	app_users-api.1	default	Running	Starting less than a second ago
tdof9stcggkb	app_api-gateway.1	default	Running	Running 4 seconds ago
u0h3dkli6dsy	app_statistics-db.1	default	Running	Running 9 seconds ago
7lofn2xgqu85	app_documents-db.1	default	Running	Running 11 seconds ago
ospknm7f99sd	app_users-db.1	default	Running	Running 12 seconds ago
5tfhl1vl84ua	app_statistics-api.1	default	Running	Running 13 seconds ago
kvrfbl5lixai	app_documents-api.1	default	Running	Running 16 seconds ago
sfsxngk9dlkx	app_users-api.2	default	Running	Starting less than a second ago
m1v8ccxxyoh6	app_statistics-api.2	default	Running	Running 13 seconds ago
jbq8m2tmeca8	app_documents-api.2	default	Running	Running 16 seconds ago
mzin8tkt711l	app_users-api.3	default	Running	Starting less than a second ago
w2d82isp1g9s	app_statistics-api.3	default	Running	Running 13 seconds ago
lphdcl3rx1sv	app_documents-api.3	default	Running	Running 16 seconds ago

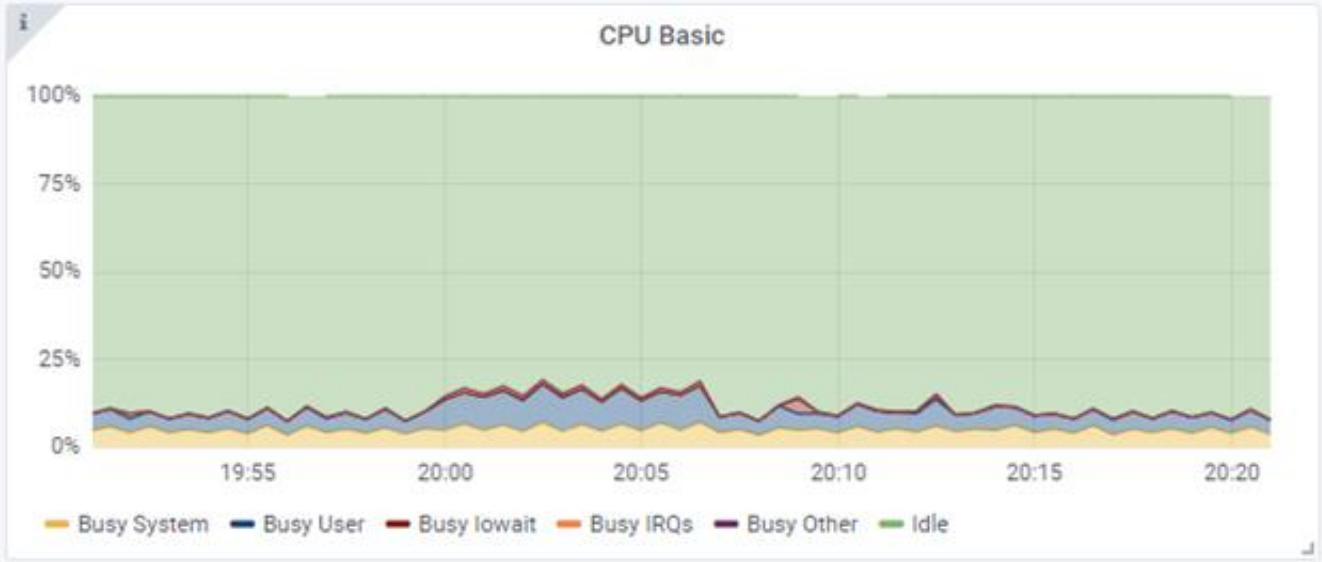
Забезпечення відмово-стійкості сервісів

ID	NAME	NODE	DESIRED STATE	CURRENT STATE
intzhwvqw282	app_users-api.1	default	Running	Starting less than a second ago
tdof9stcggkb	app_api-gateway.1	default	Running	Running 4 seconds ago
u0h3dkli6dsy	app_statistics-db.1	default	Running	Running 9 seconds ago
7lofn2xgqu85	app_documents-db.1	default	Running	Running 11 seconds ago
ospknm7f99sd	app_users-db.1	default	Running	Running 12 seconds ago
5tfhl1vl84ua	app_statistics-api.1	default	Running	Running 13 seconds ago
kvrfbl5lixai	app_documents-api.1	default	Running	Running 16 seconds ago
sfsxngk9dlkx	app_users-api.2	default	Running	Starting less than a second ago
m1v8ccxxyoh6	app_statistics-api.2	default	Running	Running 13 seconds ago
jbq8m2tmeca8	app_documents-api.2	default	Running	Running 16 seconds ago
mzin8tkt711l	app_users-api.3	default	Running	Starting less than a second ago
w2d82isp1g9s	app_statistics-api.3	default	Running	Running 13 seconds ago
lphdcl3rx1sv	app_documents-api.3	default	Running	Running 16 seconds ago

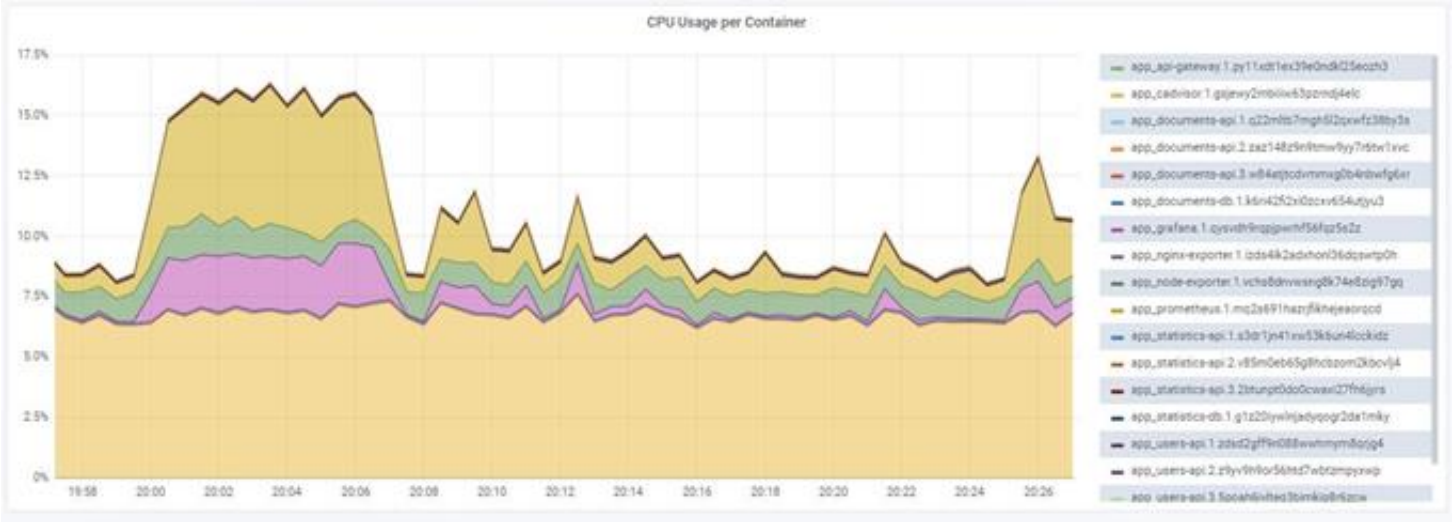
Аварійна зупинка контейнера

```
docker@default: $ docker service ls
ID                  NAME                MODE                REPLICAS
5411m9xiyhk2        app_api-gateway     replicated          1/1
3sg6xr1anyou        app_documents-api   replicated          3/3
bkhjfsgtw3bp        app_documents-db    replicated          1/1
wd7zn195n52k        app_statistics-api  replicated          3/3
49rhwychjzij        app_statistics-db   replicated          1/1
bvs54deujqy7        app_users-api       replicated          2/3
qh519471dqub        app_users-db        replicated          1/1
```

Графік використання CPU хост-машиною



Графік використання CPU контейнерами



Запит на отримання метрик

Використання RAM контейнерами

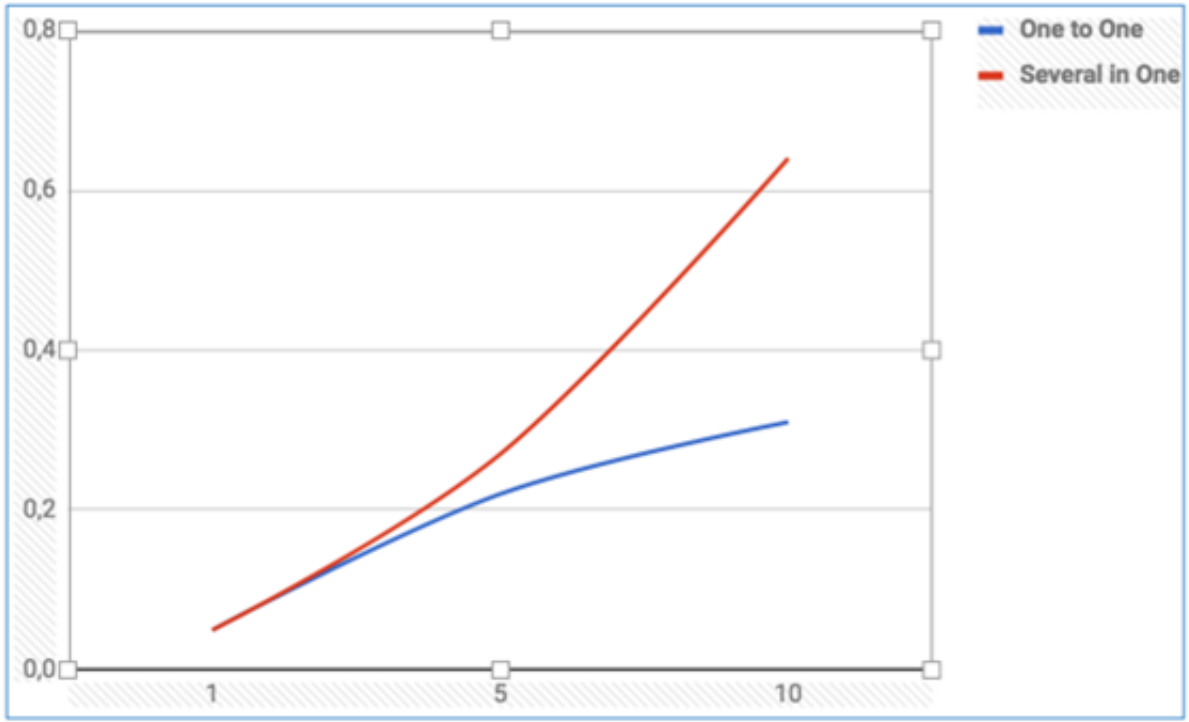
Metric	Current
app_statistics-api.2.v85m0eb65g8hcbzom2kbcvlj4	12.60 MB
app_statistics-api.1.s3dr1jn41xw53k6un4lcckidz	13.00 MB
app_prometheus.1.mq2s691hazrjfikhejeaorqcd	94.88 MB
app_node-exporter.1.vchs8dnvwsng8k74e8zig97gq	12.75 MB
app_nginx-exporter.1.izds4ik2adxhonl36dqswtp0h	6.97 MB
app_grafana.1.qysvdh9rqpjpwrhf56fqz5s2z	48.51 MB
app_documents-db.1.k6ri42fi2xi0zcxv654utjyu3	77.32 MB
app_documents-api.3.w84atjtcdvmmxg0b4nbwfg6xr	14.75 MB
app_documents-api.2.zaz148z9n9tmw9yy7r6tw1xvc	12.72 MB
app_documents-api.1.q22mltb7mgh5l2qxwfz38by3s	13.15 MB
app_cadvisor.1.gsjewy2mbiiiw63pzrndj4elc	49.65 MB
app_api-gateway.1.py11xdt1ex39e0ndkl25eozh3	2.34 MB

Кількість прийнятих з’єднань «Nginx»

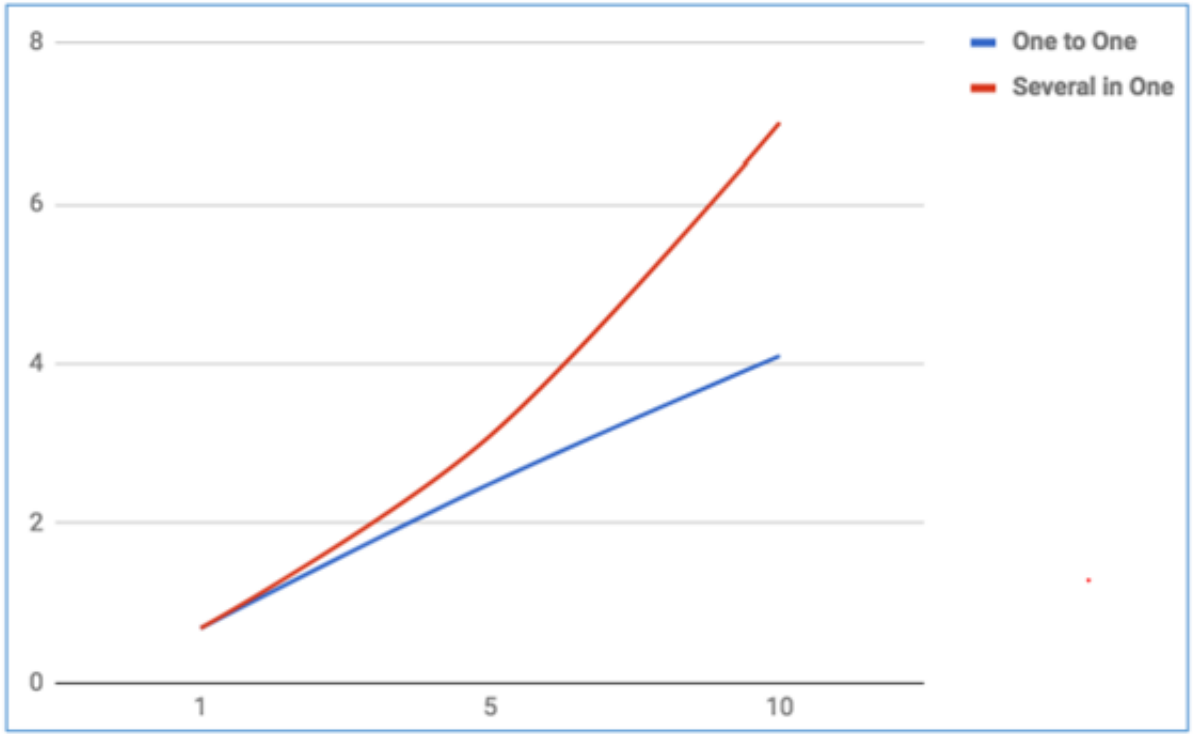
Використання «CPU and Disk» під час різних підходів до контейнеризації

№ п/п	*Підхід контейнеризації*	*1 мікросервіс*	*5 мікросервісів*	*10 мікросервісів*
1.	Один мікросервіс – один контейнер	5% CPU	22% CPU	31% CPU
		691 MB	2.5 GB	4.1 GB
2.	Усі мікросервіси в одному контейнері	5% CPU	27%CPU	64% CPU
		689 MB	3.1 GB	7 GB

Графік залежності між кількістю мікросервісів та використанням CPU



Графік залежності між кількістю мікросервісів та використанням диску



Висновки

- Проаналізовано існуючі системи управління IT-сервісами та технології контейнеризації. Визначено переваги та недоліки різних підходів, а також обґрунтовано вибір мікросервісної архітектури та Docker для розробки комплексної системи.
- Розроблено модель управління процесами для дистанційної команди. Враховано специфіку роботи IT-спеціалістів та забезпечено ефективну взаємодію між членами команди.

Висновки

- Створено комплексну систему управління IT-сервісами з використанням Docker. Реалізовано API Gateway, сервіси управління користувачами та документами, а також сервіс збору статистики. Забезпечено стандартизацію формату повідомлень та застосовано шаблони проектування для надійності та масштабованості.

Висновки

- Забезпечено відмовостійкість системи за допомогою Docker Swarm. Кластеризація дозволяє розподілити навантаження та забезпечити безперебійну роботу системи.
- Впроваджено систему моніторингу з використанням Prometheus та Grafana. Збір та візуалізація метрик продуктивності дозволяють контролювати стан системи та своєчасно реагувати на проблеми.

Висновки

- Проведено тестування та оцінено ефективність запропонованої системи. Результати демонструють підвищення продуктивності, надійності та ефективності використання ресурсів.
- Розроблена система має значний потенціал для покращення управління IT-сервісами в сучасних компаніях.

Додаток А. ВИХІДНИЙ КОД КОНФІГУРАЦІЙНОГО ФАЙЛУ DOCKER

```
version: "3.7" services:  api-gateway:
 image: api-gateway:latest deploy:  replicas: 1 ports:  - 80:80 depends_on:  - users-api
-                               documents-api
-                               statistics-api
  users-api:
 image: zhenia97chap/users-api:latest deploy:  replicas: 3 depends_on:  - users-db documents-api:
image: documents-api:latest deploy:  replicas: 3 depends_on:
-                               documents-db
  statistics-api:
 image: statistics-api:latest deploy:  replicas: 3 depends_on: - statistics-db users-db:
 image: mysql:5.7.21 deploy:  replicas: 1 environment:
 MYSQL_USER: root
 MYSQL_ROOT_PASSWORD: password MYSQL_DATABASE: users-api volumes:
-                               db_users_data:/var/lib/mysql  documents-db: image: mysql:5.7.21  deploy:  replicas: 1 environment:
 MYSQL_USER: root
 MYSQL_ROOT_PASSWORD: password MYSQL_DATABASE: documents-api volumes:
-                               db_documents_data:/var/lib/mysql  statistics-db: image: mysql:5.7.21 deploy: replicas: 1 environment:
 MYSQL_USER: root
 MYSQL_ROOT_PASSWORD: password MYSQL_DATABASE: statistics-api volumes:
-                               db_statistics_data:/var/lib/mysql
 prometheus:
 image: prom/prometheus:v2.15.2  deploy:  placement: constraints:  - node.role == manager  volumes:
-                               ./prometheus/:/etc/prometheus/  - prometheus_data:/prometheus  command:
-                               '--config.file=/etc/prometheus/prometheus.yml'
-                               '--storage.tsdb.path=/prometheus'
-                               '--web.console.libraries=/etc/prometheus/console_libraries'
-                               '--web.console.templates=/etc/prometheus/consoles'
-                               '--storage.tsdb.retention=200h'  - '--web.enable-lifecycle'  ports:  - 9090:9090  node-exporter:  image: prom/node-exporter:v0.18.1  user: root volumes:
-                               /proc:/host/proc:ro
-                               /sys:/host/sys:ro  - /:/rootfs:ro  command:
-                               '--path.procfs=/host/proc'
-                               '--path.sysfs=/host/sys'
-                               '--collector.filesystem.ignored-mount-points=^/(sys|proc|dev|host|etc)($$|/)'
  nginx-exporter:
 image: nginx/nginx-prometheus-exporter:0.5.0  environment:
-                               SCRAPE_URI=http://192.168.99.100/nginx_status
-                               TELEMETRY_PATH=/metrics  - NGINX_RETRIES=10  logging:
 driver: "json-file"  options:
 max-size: "5m"
  cadvisor:
 image: google/cadvisor:latest  volumes:
-                               /:/rootfs:ro
-                               /var/run:/var/run:rw
-                               /sys:/sys:ro
-                               /var/lib/docker/:/var/lib/docker:ro  grafana:
 image: grafana/grafana:latest  volumes:
-                               grafana_data:/var/lib/grafana  environment:
-                               GF_SECURITY_ADMIN_USER=admin
-                               GF_SECURITY_ADMIN_PASSWORD=admin - GF_USERS_ALLOW_SIGN_UP=false  ports:  - 3000:3000 volumes:  db_users_data:  driver: local  db_documents_data:
 driver: local  db_statistics_data:
driver: local  prometheus_data:  driver: local  grafana_data:  driver: local
```

Додаток Б. КОНФІГУРАЦІЯ ВЕБ-СЕРВЕРА СЕРВІСУ API-GATEWAY

```
server { listen 80;
 index index.php index.html; root /var/www/public;
 access_log /var/log/nginx/access.log; error_log /var/log/nginx/error.log; fastcgi_split_path_info ^(.+\.php)(/.+)$; fastcgi_index index.php; include fastcgi_params;
 fastcgi_param SCRIPT_FILENAME ${document_root}/index.php; fastcgi_param PATH_INFO $fastcgi_path_info;
  location / {
 return 200 'App is alive'; add_header Content-Type text/plain;
 }
 location /api/v1/users{ rewrite ^(.*)/$ $1 permanent; fastcgi_pass users-api:9000;
 }
 location /api/v1/documents{ rewrite ^(.*)/$ $1 permanent; fastcgi_pass documents-api:9000;
 }
 location /api/v1/statistics{ rewrite ^(.*)/$ $1 permanent; fastcgi_pass statistics-api:9000;
 }
}
```

Додаток В. DOCKERFILE, МІКРОСЕРВІСІВ, USERS-API, DOCUMENTS-API,STATISTICS-API

```
FROM php:7.2-fpm
RUN apt-get update \
 && apt-get -y install git \
 && apt-get -y install zip \
 && apt-get -y install procps \
 && apt-get -y install htop \
 && apt-get -y install nano \
 && apt-get -y install supervisor \
 && apt-get -y install net-tools \
 && apt-get -y install nginx
RUN docker-php-ext-install pdo_mysql
COPY . /var/www
COPY docker/supervisor/conf.d /etc/supervisor/conf.d/
COPY docker/nginx/conf.d/default.conf /etc/nginx/conf.d
RUN chown -R www-data:www-data /var/www
RUN rm /etc/nginx/sites-available/default /etc/nginx/sites-enabled/default
WORKDIR /var/www
                        RUN curl -sS https://getcomposer.org/installer | php                    -- --
installdir=/usr/local/bin --filename=composer
RUN composer install -n --prefer-dist --ignore-platform-reqs
RUN mv .env.example .env
RUN php artisan key:generate
CMD ["/usr/bin/supervisord", "-n"]
```